SOFTWARE ARCHITECTURE CS590 FINAL PROJECT

PREPARED FOR

Final Project

Anthony Sander

PREPARED BY

Jhon Edward Ramirez Brice

Luzan Baral

Sanjaya Koju

Shrawan Adhikari

Supriya Ghising

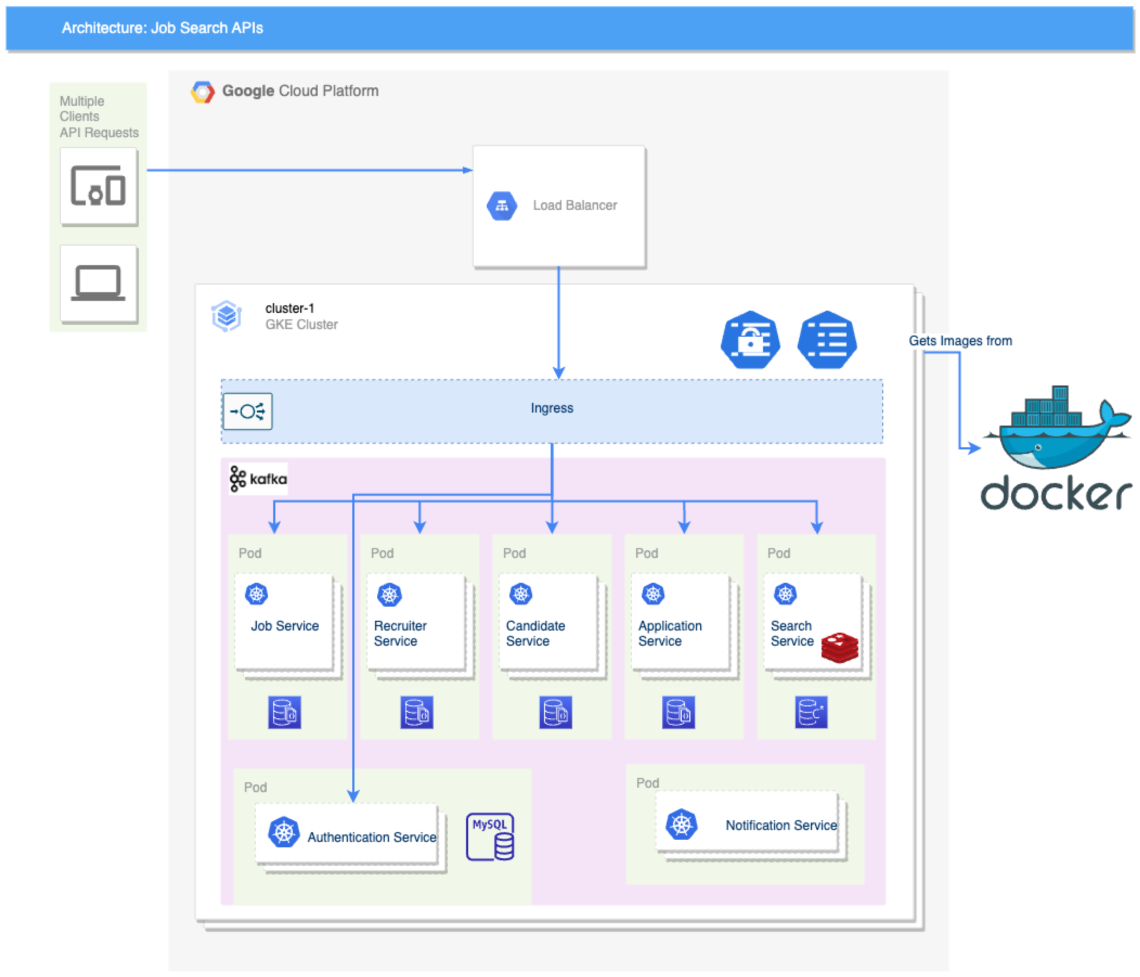
PROJECT SUMMARY

Microservice-based application developed using Spring Boot to fulfill the Job Searching application business domain. We took six essential services for this domain and developed these services as micro-service. The sole purpose of this project was to learn software architecture when it comes to building applications, writing test cases, and deploying them as microservices, alongside learning Docker, Kubernetes, Kafka, Zipkin, Redis, and MongoDB.

1. Project Overview

Microservices Application to mimic the business domain of the Job Search Application with features to create jobs, recruiters, and candidates, and search and apply for those jobs. The project is built using Spring boot following microservice architecture, and all its services are orchestrated and managed using K8s.

2. Architecture



Authentication Service

Authentication Service exposes APIs to create users and login them into the system. When a user is logged in a JWT token is created with an expiry time of 1 day and returned. This JWT token can be used to get authenticated access to other services, as it is managed accordingly in the system.

Job Service

Job Service exposes APIs to create and read jobs. Whenever a user creates a job it is stored in MongoDB and the service also produces a message to Kafka. This message in Kafka is consumed by the consumer in the Search service and is stored in Elasticsearch.

Recruiter Service

Allows registered users to create a profile as a recruiter, this an access which will have access to candidates with premium features. Also, this profile will allow one to get information about a recruiter and know the list of recruiters, as well as the company or companies recruiters, work for.

Candidate Service

Candidates can apply for the jobs if they are registered in the system. They are also able to create their profile likes: skillsets, experience, etc. Whenever candidates create their profile in the system. They received a notification via mail that profiles were created in the system.

Notification Service

Notification service consumes the message created while creating a candidate profile and a new job is created by the company. Through this consume message information it will send the notification mail to the respective person and organization.

Search Service

Search Service is used to search for Jobs and candidates, by consuming the Kafka topics. This service uses Elasticsearch to search and results are stored in the Redis cache. Redis was our choice for implementing a highly available in-memory cache to decrease data access latency, increase throughput, and ease the load off your relational or NoSQL database and application.

Application Service

This is the service developed to facilitate the job application process from our job portal system. As the search service provides the information of the job details and the customer details in the service, the logged-in candidate can apply for a job which will surf through the application service. Then the job validity is checked whether the valid candidate has applied or not and also checks whether the applied job is valid or not. For this, we approach the asynchronous process of getting information about the job's existence. We used Kafka to connect to the job service to know the job validity through which we can get the response status to apply. As to the response, if the job is found and valid, the candidate can apply. Throughout this process ongoing in the system, the candidate will be notified with the message "checking job. Please try again later".

3. Use of Kafka

Kafka is used as a center to collect messages during the creation of jobs and candidates, which is done by the producers inside the individual services. These produced events are consumed by the search service and stored in the in-service database. This in-service database is later used by the search service when searching for jobs or candidates.

4. Selection of Databases

MongoDB

Data is highly available with MongoDB as it makes multiple copies of the same data and sends copies of data across different servers. In case any server fails, data can be retrieved from another server without delay.

As MongoDB is partition tolerant and also highly consistent according to CAP theorem, we decided to use MongoDB on most of our services.

MySQL

We have used MySQL with our authentication service as we saw a possible future for the necessity of relational data for user information in this particular service. Apart from that, we also thought that authentication services needed to be highly available and consistent.

ElasticSearch

We have used ElasticSearch in our search service to make searching faster. ElasticSearch allows you to store, search, and analyze huge volumes of data quickly and in near real-time and give back answers in milliseconds. It's able to achieve fast search responses because instead of searching the text directly, it searches an index.

5. Kubernetes

We have used Kubernetes for a container orchestration system to automate the management, scaling, and deployment of our microservice application. Along with that Kubernetes made it easier to manage tools like config secrets, sleuth, Zipkin, and ingress. We have also used Helm to install the required tools.

6. Monitoring and Tracing

We have used Zipkin for distributed tracing in our local deployment.

We have also used Grafana and Prometheus for monitoring the services, but so far we could only test them in our local deployment.

7. CI/CD

In our project, we have tried and tested some of the basic ways of implementing CI/CD tools, through which we can observe the progress once the commit is made on the main branch of the project on GitHub.

8. Obstacles

One of the major obstacles was the technology itself, as it was new for everyone, and there was a learning curve on all the services and features used, provided a week-long time.

Moving to Google Cloud - Google Kubernetes Engine was also a major technical obstacle. While we didn’t have issues deploying to Kubernetes locally, deploying it to GKE became a nightmare as we encountered new errors and had to work intensively to solve them.

9. Conclusion

This project allowed us to learn and implement almost everything related to microservice architecture, which we discussed as a concept in the classroom. In addition, we also experienced working in a team and exploring new technologies over a week-long period.